Shipping Multiple Devicetrees

How to Identify Which DTB is for my Board?

Elliot Berman <quic_eberman@quicinc.com> (Senior Engineer)

Qualcomm Innovation Center, Inc.

About

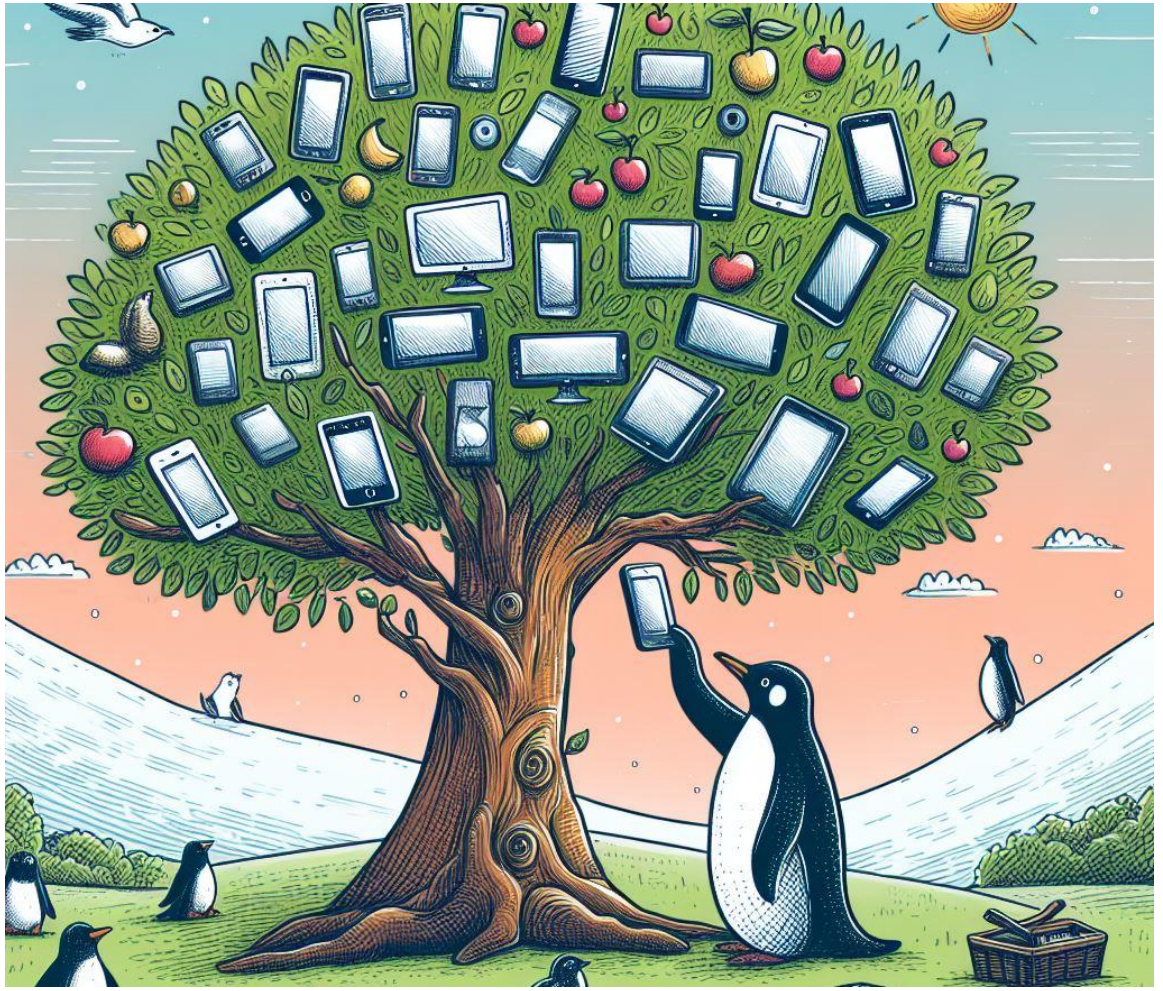
- Linux kernel developer at Qualcomm Innovation Center, Inc. in San Diego, CA
- Some things I've worked on:
  - Gunyah hypervisor
  - Enable Android's Generic Kernel Image (GKI) @ Qualcomm
  - Android's kernel/build

How Devicetrees are Loaded

- Kernel* authors devicetrees
- Bootloader picks up the devicetree and passes to kernel
  - OR: devicetree is appended to kernel
- Kernel doesn't own the bootloader, leaves it to ecosystem to package and pick the DTB it wants

- Ideally, DTB comes from firmware but reality is most boards use kernel-authored devicetree

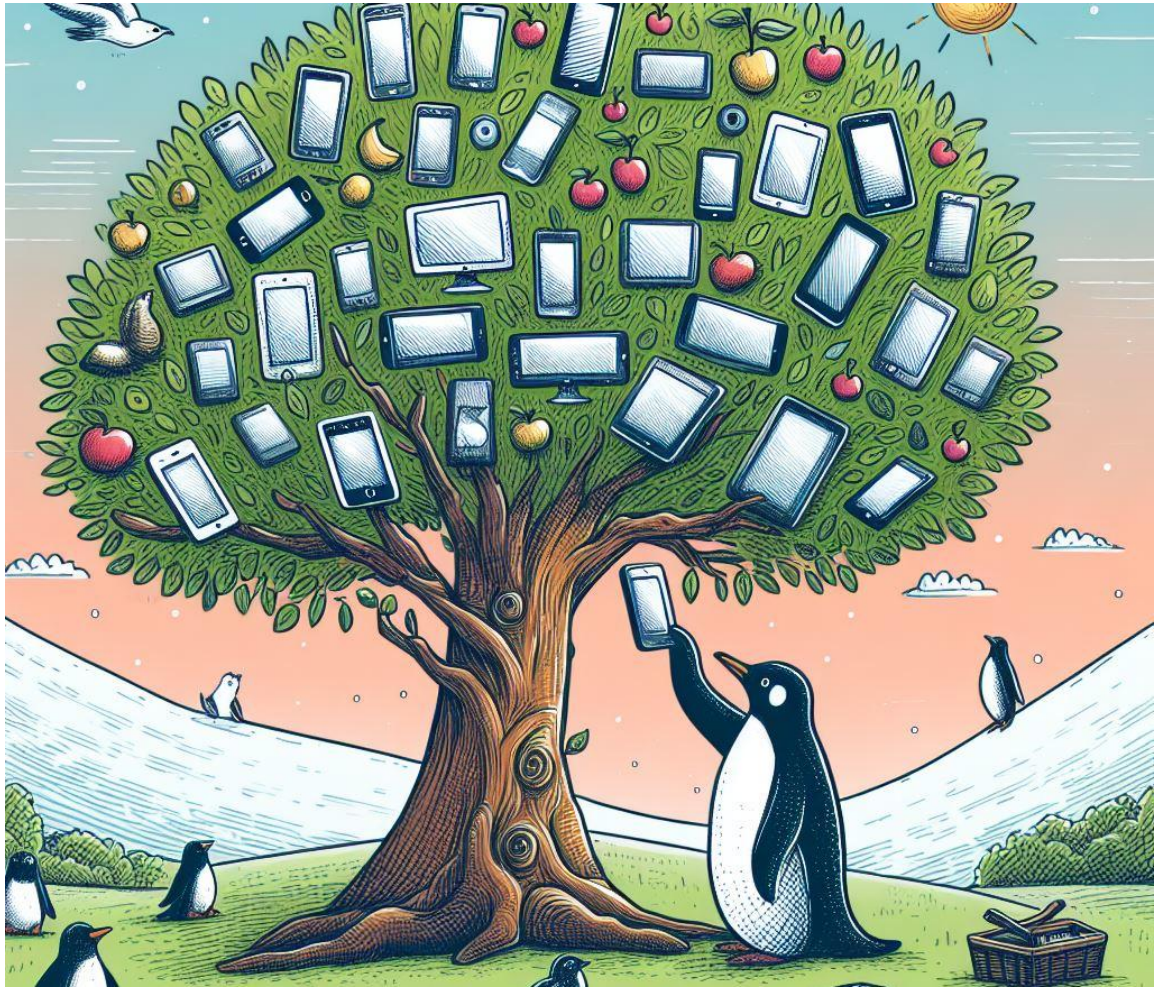
*kernel = Linux kernel, Zephyr, FreeBSD, more



How Devicetrees are Loaded

- Kernel* authors devicetrees
- Bootloader picks up the devicetree and passes to kernel
  - OR: devicetree is appended to kernel
- Kernel doesn't own the bootloader, leaves it to ecosystem to package and pick the DTB it wants

Maybe just one DTB
that developer describes
in the build command



*kernel = Linux kernel, Zephyr, FreeBSD, more

How Devicetrees are Loaded

- Kernel* authors devicetrees
- Bootloader picks up the devicetree and passes to kernel
  - OR: devicetree is appended to kernel
- Kernel doesn't own the bootloader, leaves it to ecosystem to package and pick the DTB it wants

Maybe just one DTB
that developer describes
in the build command

mkbootimg –d qrb5165-rb5.dtb

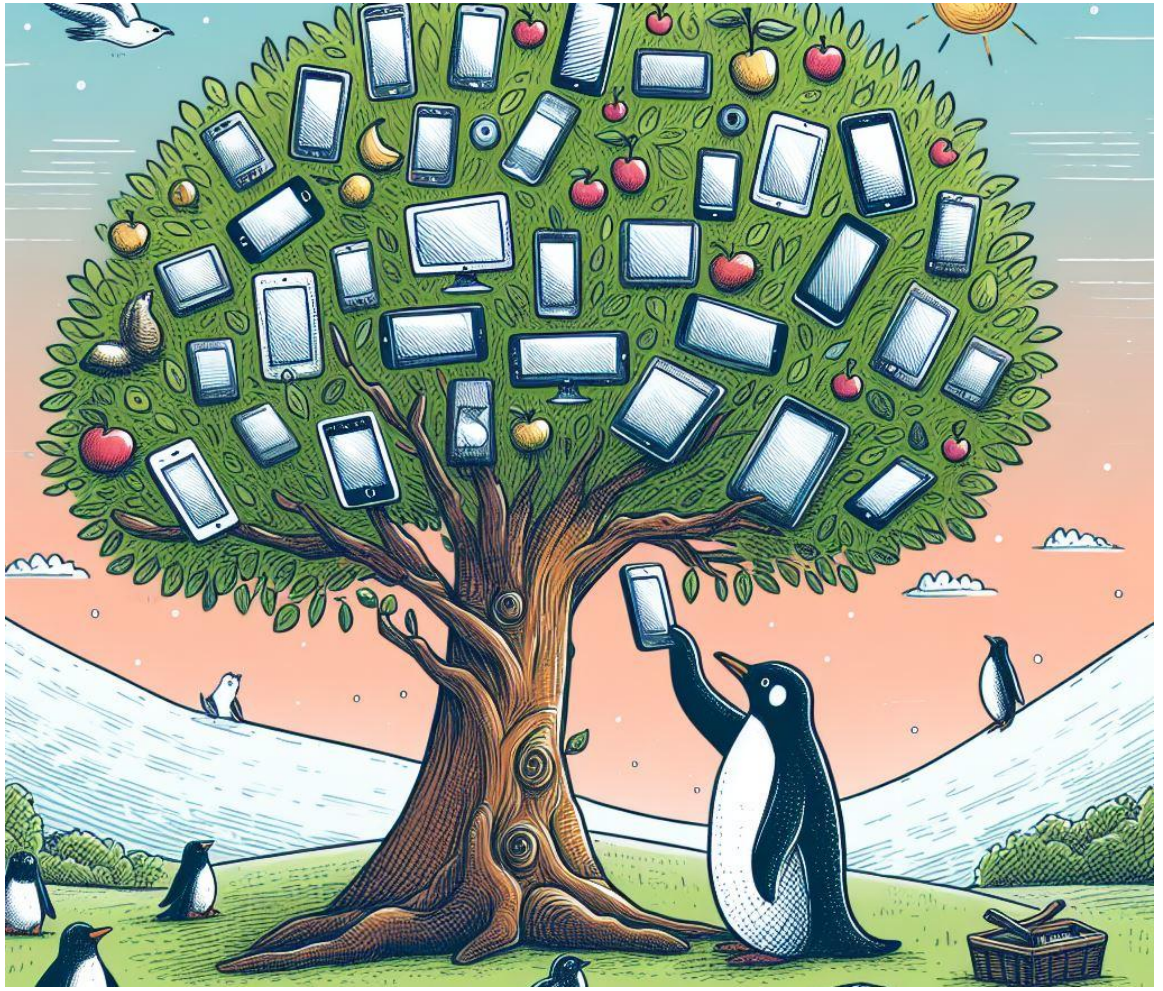
cat Image qrb5165-rb5.dtb > Image_dtb

*kernel = Linux kernel, Zephyr, FreeBSD, more

How Devicetrees are Loaded

- Kernel* authors devicetrees
- Bootloader picks up the devicetree and passes to kernel
  - OR: devicetree is appended to kernel
- Kernel doesn't own the bootloader, leaves it to ecosystem to package and pick the DTB it wants

Maybe multiple DTBs get packaged and bootloader must pick

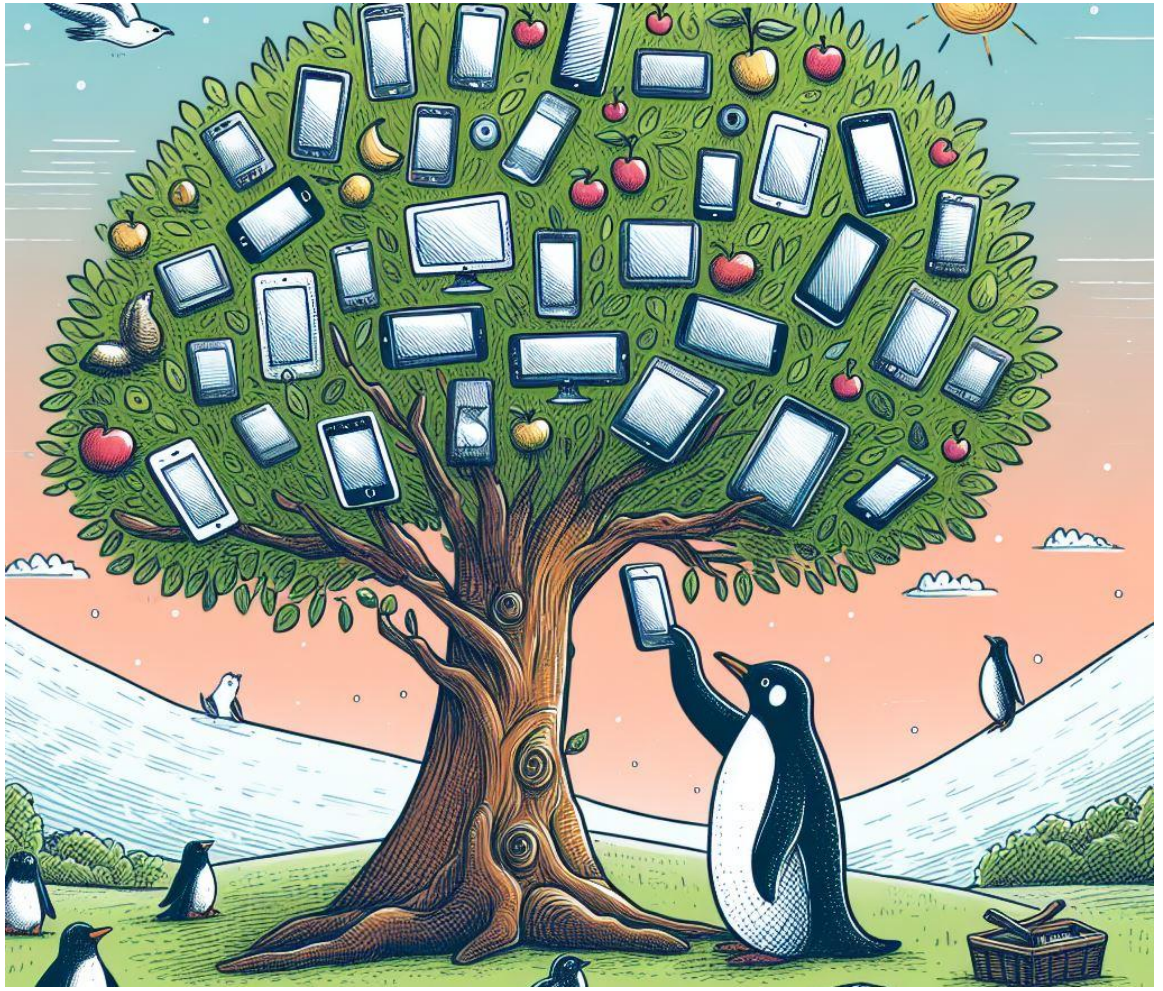


*kernel = Linux kernel, Zephyr, FreeBSD, more

How Devicetrees are Loaded

- Kernel* authors devicetrees
- Bootloader picks up the devicetree and passes to kernel
  - OR: devicetree is appended to kernel
- Kernel doesn't own the bootloader, leaves it to ecosystem to package and pick the DTB it wants

Focus of this talk is on **picking** the right DTB
not a universal packaging mechanism (future?)



*kernel = Linux kernel, Zephyr, FreeBSD, more

Shipping Multiple Devicetrees

- Device manufacturers want to ship one binary to variety of devices
  - OEMs: single software package that supports phone A and phone A Pro
  - Qualcomm: single software package that supports multiple reference devices
- Reduce friction when receiving a new board

Shipping Multiple Devicetrees

- Boards could have different
  - sensors
  - displays
  - coprocessor attachments
    - GPU
    - modem
    - Wi-Fi
  - PMICs

Devices from Qualcomm Technologies, Inc.

- Read SoC's HW identifier registers
- Read configuration data table (CDT)
  - Answers "which reference board is this?"
  - HDK, Robotics Dev Kit, Auto Dev Platform
  - What else attached?
    - Vision kit?
    - Display?
    - PMIC?
  - Some hardware features are encoded in board version, some are separate identifiers
- 1 software package: ~20 DTBs, ~100 overlays
- SKU is too much information
  - SKU encodes DRAM manufacturer and more
  - Demuxing SKU requires unnecessary s/w updates

Devices from Qualcomm Technologies, Inc.

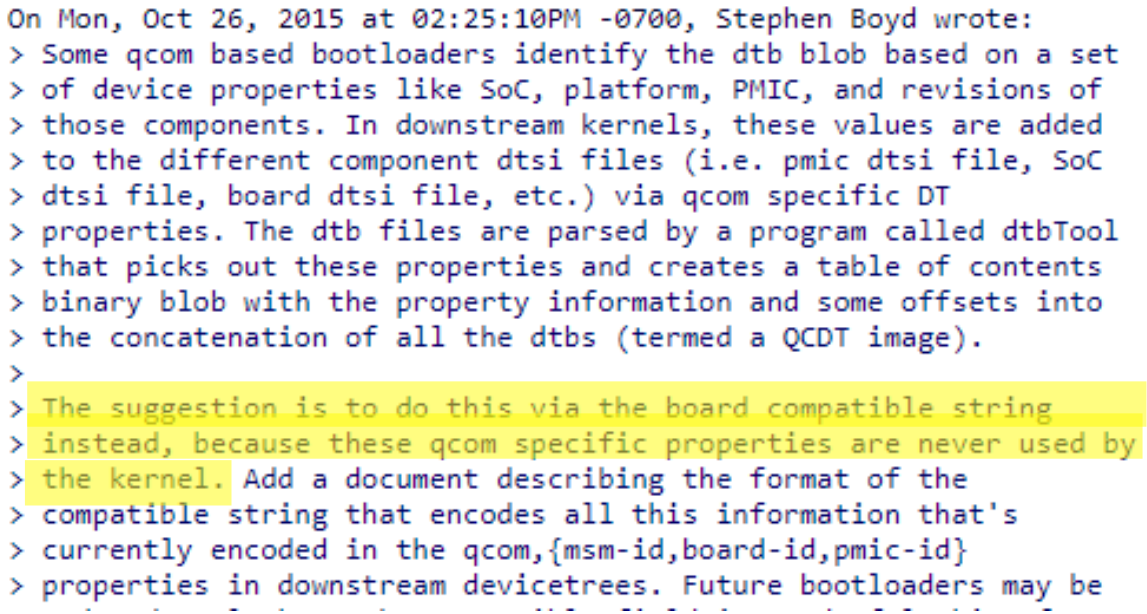
- Bootloader reads SoC HW identifiers, CDT
- Android Boot Loader [1,2] maps the values into “qcom,msm-id”, “qcom,board-id”, “qcom,pmic-id”, “qcom,oem-id”

DT property	Individual fields
qcom,msm-id	
	Chipset Id
	SoC Revision
qcom,board-id	
	Board Id
	Board Major
	Board Minor
	Subtype
	DDRtype
	BootDevice Type
qcom,pmic-id	
	Slave Id
	PMIC Id
	PMIC Major
	PMIC Minor
qcom,oem-id	
	OEM Id

[1] https://git.codelinaro.org/clo/la/abl/tianocore/edk2/-/blob/KERNEL.PLATFORM.2.0.r1-17200-kernel.0/QcomModulePkg/Library/BootLib/LocateDeviceTree.c?ref_type=tags
[2] https://lore.kernel.org/all/1710418312-6559-1-git-send-email-quic_amrianan@quicinc.com/

A decade later, we haven’t solved this

- msm-id refused because they aren’t used by the kernel [1]



```
On Mon, Oct 26, 2015 at 02:25:10PM -0700, Stephen Boyd wrote:
> Some qcom based bootloaders identify the dtb blob based on a set
> of device properties like SoC, platform, PMIC, and revisions of
> those components. In downstream kernels, these values are added
> to the different component dtsi files (i.e. pmic dtsi file, SoC
> dtsi file, board dtsi file, etc.) via qcom specific DT
> properties. The dtb files are parsed by a program called dtbTool
> that picks out these properties and creates a table of contents
> binary blob with the property information and some offsets into
> the concatenation of all the dtbs (termed a QCDT image).
>
> The suggestion is to do this via the board compatible string
> instead, because these qcom specific properties are never used by
> the kernel. Add a document describing the format of the
> compatible string that encodes all this information that's
> currently encoded in the qcom,{msm-id,board-id,pmic-id}
> properties in downstream devicetrees. Future bootloaders may be
```

[1]: https://lore.kernel.org/lkml/20151106201504.GA18276@qualcomm.com

A decade later, we haven’t solved this

- msm-id refused because they aren’t used by the kernel [1]
- Tried documenting/using compatible strings

[1]: https://lore.kernel.org/lkml/20151106201504.GA18276@qualcomm.com

A decade later, we haven't solved this

- msm-id refused because they aren't used by the kernel [1]
- Tried documenting/using compatible strings
  - It didn't catch on
  - Compatible strings get mind-swimmingly long
  - Requires custom parsing

```
compatible = "qcom,<SoC>(-<soc_version>)(-<foundry_id>)
              -<plat_type>(/<subtype>)(-<plat_version>)
             (-<mb>MB)(-<panel>-panel)(-boot-<boot>)(-<pmic>(-v<pmic_version>)){0-4}"
```

```
om,apq8074-v2.0-2-dragonboard/1-v0.1-512MB-panel-qHD-boot-emmc_sdc1-pm8941-v0.2-pm8909-v2.
```

[1]: https://lore.kernel.org/lkml/20151106201504.GA18276@qualcomm.com

A decade later, we haven't solved this

- msm-id refused because they aren't used by the kernel [1]
- Tried documenting/using compatible strings
  - It didn't catch on
  - Compatible strings get mind-swimmingly long
  - Hard to write, hard to parse
- Tried encoding the information outside the DTB into a DT image
  - QCDT image transformed a DTB with msm-id into a DTB with some extra metadata at the header
  - No clear advantage why this is better than leaving + using msm-id

[1]: https://lore.kernel.org/lkml/20151106201504.GA18276@qualcomm.com

Upstream Approaches

- Use the compatible string
  - Requires string parsing which can quickly get complex
  - Requires porting the parsing to bootloaders that needs to choose the DT
  - Used by Chromebook, Tegra (others?)
  - Compatible strings often include the SoC, but board X DTB isn't going to work on board Y, even if they have the same SoC (st,stm32mp157; qcom,sm6115)
- Put everything in the DT + bootloader picks what to enable
  - Creates contract between bootloader and devicetree

Downstream approaches

- Qualcomm: msm-id and board-id
  - Bitfield containing magic constants from board ROM
- Google Pixel: soc_id (similar approach to Qualcomm)
- AOSP: dtbo.img metadata
  - Adds optional id, revision, and u32 custom[4] for each DTB
- U-boot/FIT: Uses filename to match DTB
- Coreboot/FIT: configuration based on vendor,mainboard + board_id + sku_id

DTB Selection Feature Wishlist

- ✓Lower the effort to support new boards
- ✓Expect HW to provide more info than S/W may care about
  - ✓We probably don’t care about rev 2.1 of QCM6490 SoC
- ✓Translatable to different DTB packaging formats
- ✓Documented mechanism for how to match properties
  - ✓Consistent behavior from bootloader A to bootloader B

Current concept

Proposal; no bootloaders support this scheme today

SM8650 v2 with HDK v1.1 or HDK v1.0

```
#include <dt-bindings/arm/qcom,ids.h>

/ {
  board-id {
    qcom,soc-version = <QCOM_ID_SM8650 QCOM_SOC_REVISION(2)>;
    qcom,board-version = <QCOM_BOARD_ID(HDK, 1, 0)>,
                         <QCOM_BOARD_ID(HDK, 1, 1)>;
  };
};
```

Google Pazquel360 with LTE (newest rev)

```
/ {
  compatible = "google,pazquel-sku22", "google,pazquel-sku20",
               "qcom,sc7180";
  // Equivalent representation:
  board-id {
    google,board = "pazquel";
    google,sku = <22>, <20>;
  };
};
```

Current concept

SM8650 v2 with HDK v1.1 or HDK v1.0

```
#include <dt-bindings/arm/qcom,ids.h>

/ {
  board-id {
    qcom,soc-version = <QCOM_ID_SM8650 QCOM_SOC_REVISION(2)>;
    qcom,board-version = <QCOM_BOARD_ID(HDK, 1, 0)>,
                         <QCOM_BOARD_ID(HDK, 1, 1)>;
  };
};
```

Google Pazquel360 with LTE (newest rev)

```
/ {
  compatible = "google,pazquel-sku22", "google,pazquel-sku20",
               "qcom,sc7180";
  // Equivalent representation:
  board-id {
    google,board = "pazquel";
    google,sku = <22>, <20>;
  };
};
```

Bootloader query the platform abstraction for “qcom,soc” value and compare if any value matches.

Current concept

SM8650 v2 with HDK v1.1 or HDK v1.0

```
#include <dt-bindings/arm/qcom,ids.h>

/ {
  board-id {
    qcom,soc-version = <QCOM_ID_SM8650 QCOM_SOC_REVISION(2)>;
    qcom,board-version = <QCOM_BOARD_ID(HDK, 1, 0)>,
                         <QCOM_BOARD_ID(HDK, 1, 1)>;
  };
};
```

Google Pazquel360 with LTE (newest rev)

```
/ {
  compatible = "google,pazquel-sku22", "google,pazquel-sku20",
               "qcom,sc7180";
  // Equivalent representation:
  board-id {
    google,board = "pazquel";
    google,sku = <22>, <20>;
  };
};
```

Bootloader query the platform abstraction for “qcom,soc” value and compare if any value matches.

If every property has a match, this is the right board

Current concept

SM8650 v2 with HDK v1.1 or HDK v1.0

```
#include <dt-bindings/arm/qcom,ids.h>

/ {
  board-id {
    qcom,soc-version = <QCOM_ID_SM8650 QCOM_SOC_REVISION(2)>;
    qcom,board-version = <QCOM_BOARD_ID(HDK, 1, 0)>,
                         <QCOM_BOARD_ID(HDK, 1, 1)>;
  };
};
```

Google Pazquel360 with LTE (newest rev)

```
/ {
  compatible = "google,pazquel-sku22", "google,pazquel-sku20",
               "qcom,sc7180";
  // Equivalent representation:
  board-id {
    google,board = "pazquel";
    google,sku = <22>, <20>;
  };
};
```

maybe UEFI protocol?

Bootloader query the platform abstraction for “qcom,soc” value and compare if any value matches.

If every property has a match, this is the right board

Current concept

SM8650 v2 with HDK v1.1 or HDK v1.0

```
#include <dt-bindings/arm/qcom,ids.h>

/ {
  board-id {
    qcom,soc-version = <QCOM_ID_SM8650 QCOM_SOC_REVISION(2)>;
    qcom,board-version = <QCOM_BOARD_ID(HDK, 1, 0)>,
                         <QCOM_BOARD_ID(HDK, 1, 1)>;
  };
};
```

Google Pazquel360 with LTE (newest rev)

```
/ {
  compatible = "google,pazquel-sku22", "google,pazquel-sku20",
               "qcom,sc7180";
  // Equivalent representation:
  board-id {
    google,board = "pazquel";
    google,sku = <22>, <20>;
  };
};
```

Could be copied into FIT configuration

Mailing list

- RFCs: Add board-id support for multiple DT selection
  - v1: https://lore.kernel.org/all/1705749649-4708-1-git-send-email-quic_amrianan@quicinc.com/
  - v2: https://lore.kernel.org/all/1710418312-6559-1-git-send-email-quic_amrianan@quicinc.com/
- Feedback so far: get a community consensus!

Current concept

Proposal; no bootloaders support this scheme today

SM8650 v2 with HDK v1.1 or HDK v1.0

```
#include <dt-bindings/arm/qcom,ids.h>

/ {
  board-id {
    qcom,soc-version = <QCOM_ID_SM8650 QCOM_SOC_REVISION(2)>;
    qcom,board-version = <QCOM_BOARD_ID(HDK, 1, 0)>,
                         <QCOM_BOARD_ID(HDK, 1, 1)>;
  };
};
```

Google Pazquel360 with LTE (newest rev)

```
/ {
  compatible = "google,pazquel-sku22", "google,pazquel-sku20",
               "qcom,sc7180";
  // Equivalent representation:
  board-id {
    google,board = "pazquel";
    google,sku = <22>, <20>;
  };
};
```

```
#include <dt-bindings/arm/qcom,ids.h>
/ {
  board-id {
    qcom,soc = <QCOM_ID_MSM8916>;
    qcom,board-version = <QCOM_BOARD_ID(HDK, 1, 0)>;
    qcom,storage = <QCOM_STORAGE_ID_EMMC>; // QCOM_STORAGE_ID_NAND
  };
};
```

Thank you

Qualcomm

Follow us on:

For more information, visit us at:

qualcomm.com & qualcomm.com/blog